University of Saskatchewan
Department of Computer Science

Cmpt 340

Final Examination

December 12, 1996

Time: 3 hours
Total Marks: 90

Professor: A. J. Kusalik
Closed Book†

Name: _____

Student Number: _____

Directions:

Answer each of the following questions in the space provided in this exam booklet. If you must continue an answer (e.g. in the extra space on the last page, or on the back side of a page), make sure you clearly indicate that you have done so and where to find the continuation.

Ensure that all written answers are legible; no marks will be given for answers which cannot be deciphered. Where a discourse or discussion is called for, please be concise and precise. If you find it necessary to make any assumptions to answer a question, please state the assumption with your answer.

Marks for each major question are given at the beginning of the question. There are a total of 90 marks.

The examination begins on the next page. Good luck.

For marking use only:

A.	_____/10	E.	_____/16	I.	_____/6
B.	_____/6	F.	_____/4	J.	_____/6
C.	_____/15	G.	_____/10		
D.	_____/12	H.	_____/5		

Total: _____/90

† Closed book, except for one optional 8.5x11 inch quick reference sheet ("cheat sheet") of the student's own compilation.

A. ***(1 mark each, 10 marks total)***
Each of the following questions requires a very short, concise, and precise answer.

1. What programming language has dominated business applications over the past 25 years?

2. What programming language was the first to support the three fundamental features of object-oriented programming?

3. What syntax description language (or formalism) was designed to describe the syntax of Algol 60?

4. Consider the following Prolog program:

```
pred( true ).
pred( (G1,G2) ) :- pred( G1 ), pred( G2 ).
pred( G ) :-   before( G ),
               clause( G, B ),
               pred( B ),
               after( G ).

before( G ) :- write( 'call' ), write( G ), nl.
before( G ) :- write( 'fail' ), write( G ), nl, fail.

after( G ) :- write( 'succeed' ), write( G ), nl.
after( G ) :- write( 'redo' ), write( G ), nl, fail.
```

   What kind of program is the program above? I.e. if asked to categorize the program above, what would you call it?

5. Name one type of static semantics. I.e. name one type of information that is part of the static semantics of a program.

6. What is the name typically given (in the context of programming languages) to the following symbol: '⊥' ?

7. Is an array data structure in a traditional procedural language more like a Gofer list, or more like a Gofer tuple?

8. The C operators '&&' and '||' are "short circuit" versions of the boolean *and* and *or* operators, respectively. That is, they evaluate their second argument only if it is necessary. This is similar to a feature / concept commonly found in functional languages. What is that concept?

9. Is the following Java class definition part of an application (application program) or an applet?

```
import java.applet.*;
import java.awt.*;

public class Mystery extends Applet {
   public static void main( String args[] ) {
      Applet applet = new Mystery();
      Frame frame = new AppletFrame( "mystery", applet, 300, 300 );
   }
}
```

10. Is it true or is it false that all of the languages studied in class (Prolog and CLP, Gofer, Prograph, and Java) have automatic garbage collection as part of their runtime system.

B. ***(2 marks each, 6 marks total)***
Each of the following questions requires a short, concise answer.

1. In a logic inference rule such as

$$\frac{E \Rightarrow E_1}{E \text{ '*' } E_2 \Rightarrow E_1 \text{ '*' } E_2}$$

which part (what) is the premise, and which part is the conclusion?

2. What is the main usefulness of axiomatic semantics? I.e. in what role has axiomatic semantics met with the best success?

3. What are the two kinds of statements that populate a Prolog database? I.e. what are the two types of clauses in a Prolog program?

C. (3 marks each, 15 marks total)

Answer each of the following questions with a concise answer.

1. A program unit is history-sensitive if it can produce different results when activated twice with the same values of parameters and accessible nonlocal variables. Would you expect a modern functional language to have history-sensitive functions? Why or why not?

2. Consider a function `make_double` defined as follows (in a functional notation, but not necessarily Gofer):

```
make_double f = (λx. f x x)
```

I.e. `make_double` is a function of one argument (and that argument is again a function). Written as a lambda expression, the function named by `make_double` is `(λf.(λx. f x x))` or simply `λf.λx.f x x`.

If $*$ is the integer multiplication operator, give the <u>type</u> of (the type expression for) the function

```
(make_double *)
```

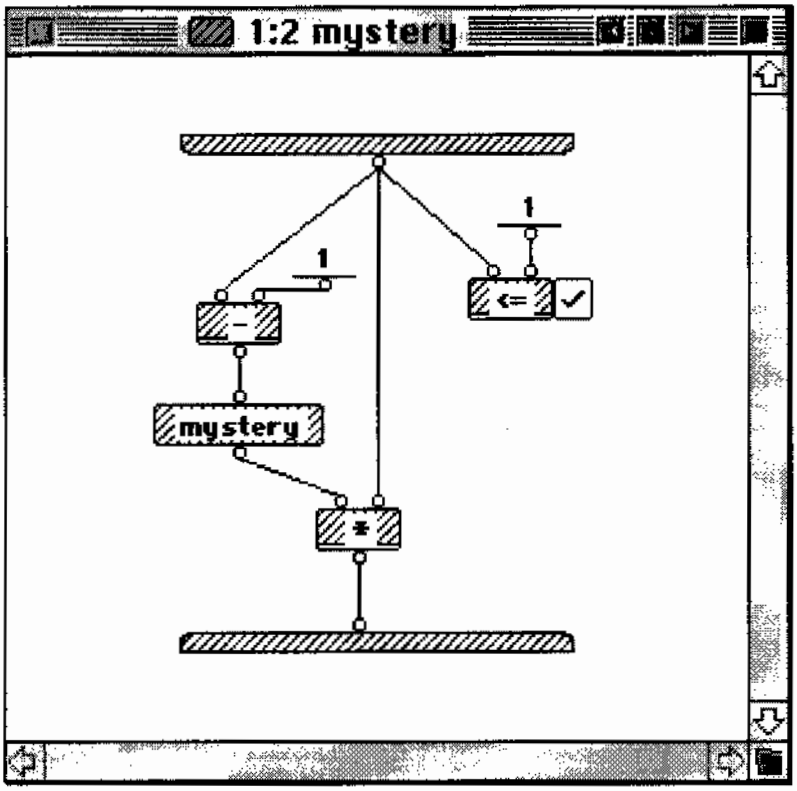
3. What relation does the following Prolog predicate p/2 compute? I.e. what relation does it name? That is, if you had to give the predicate p a more descriptive name, what would it be?

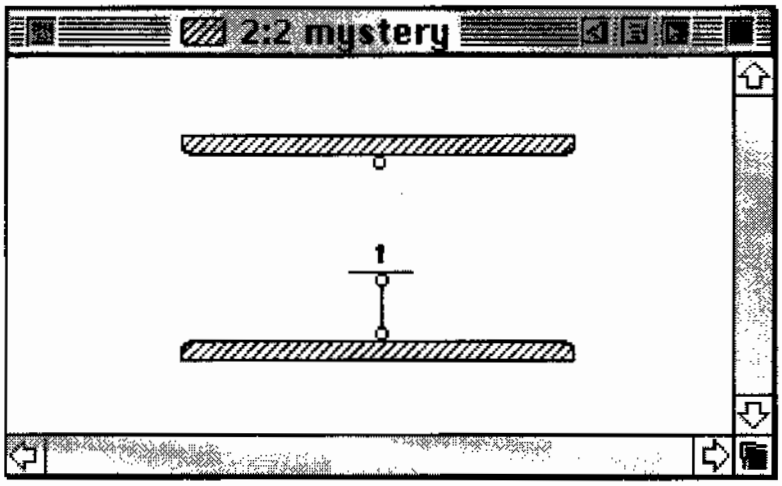
```
p( L, X ) :-
   append( _, [X], L ).
```

Assume that append/3 is defined as usual.

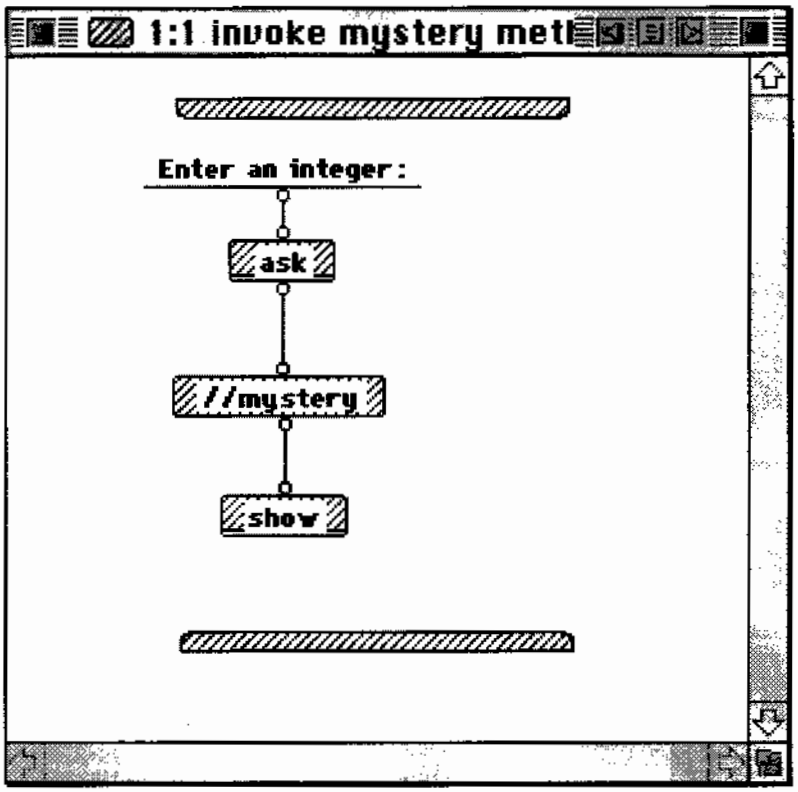
4. Normally, individual constraint logic programming languages are signified by a notation CLP(d) where d is substituted by a specific symbol or symbols (e.g. CLP(R), CLP(B), or CLP(Q,R)). What does the d in CLP(d) signify? What does it stand for? What does it mean when a specific symbol, say R, is substituted for the d?

5. Consider a Prograph method `mystery` defined as shown in the following two pictures:





The method is invoked via the following auxiliary method:



What function does the method `mystery` compute?

D. ***(3 marks each, 12 marks total)***
Give the name of a language studied in class (i.e. Gofer, Prolog and CLP, Prograph, and Java) which answers each of the following. In each case, explain and justify your answer.

Note: consider only the implementations used in class.

1. Which language studied in class would be judged as having the most well-developed programming environment? Why?

2. Which language studied in class has the weakest form of typing ("typing" as in "data types")? Why?

3. Name a language studied in class which supports implicitly dynamic variables. Justify your answer.

4. Name a language studied in class which does not support global variables ("global variables" in the Pascal, FORTRAN, or C sense). Justify your answer.

E. ***(4 marks each, 16 marks total)***
Answer each of the following questions with a concise answer.

1. Given the following BNF

```
<exp>  ::= ( <list> ) | a
<list> ::= <list> , <exp> | <exp>
```

where <exp> is the "start symbol", draw the parse tree for

```
((a,a),a,(a))
```

2. Consider the following program in a (statically-scoped) Pascal-like or C-like language:

```
program main( input, output )
   var i, j, k, m;

   procedure Q( i: integer; m: integer );
      begin
      i := i + k;
      m := j + 1
      end;

   procedure P( i: integer; j: integer );
      var k: integer;
      begin
      k := 4;
      i := i + k;
      j := j + k;
      Q( i, j );
      end;

   begin
   i := 1;
   j := 2;
   k := 3;
   P( i, k );
   writeln( i, j, k );
   end.
```

Give the three numbers printed in the case that parameters are passed (a) by value, (b) by reference, (c) by value-result, and (d) by name.

3. Suppose a specification of denotational semantics contains the statement

   $D[['0']] = 0$

   What does this signify? I.e. explain all the notation in the above statement.

4. What are two different, nontrivial properties or characteristics that Java and Prograph have in common? I.e., identify two significant, high-level similarities between Java and Prograph.

F. (4 marks)

1. Why does the word `FALSE` never print in the sequence below? (The language used is a language similar in syntax to C or Java. Assume variables `x` and `y` have already been declared, and some function or method called `print` has already been appropriately defined.)

```
x = 2;
y = 5;
if ( x = y )
   print( "TRUE\n" );
else
   print( "FALSE\n" );
```

2. The example above demonstrates the violation of a particular programming language criterion. What criterion is that? Why or how does the example above violate that criterion?

G. (10 marks)

There are other implementation techniques for accessing variables in block-structured, statically-scoped languages besides that using a static chain and static distance. One such method is called "shallow binding". Each procedure is statically allocated one copy of its activation record. This static copy always holds the information and local variables for the most recent activation of that procedure. Since the location of these activation records can be bound at compile time, variable access is always efficient. However, supporting recursion is more difficult. When a procedure is invoked, the (previous) contents of the static activation record are pushed onto a stack so that the activation record area can be used by the new procedure instance. When the callee returns, the contents of the activation record are restored from the stack.

1. Describe how a variable access would be performed/implemented under the shallow binding technique. What computations must the compiler perform? What operations must be performed at runtime?

2. Compare the shallow binding and static chain techniques for efficiency (at runtime). State which is more efficient for operations such as variable access (both within the local context and in an outer context), and procedure call and return (both recursive and non-recursive), and why. In your justification consider, for example, the number of memory references for such operations.

3. In contrast to shallow binding, the technique (covered in class) using a static chain and static binding could be called "deep binding". Explain why this name is justified; i.e. why could it warrant this name?

H. ***(5 marks)***

Consider the following statement:

> In operational semantics, an environment must be given before an abstract machine can perform any reductions. Thus one (extremely abstract) view of operational semantics is as a function
>
> Φ:Program X Environment $\rightarrow$ Environment
>
> In this sense, denotational semantics can be viewed as a Curried version of operational semantics.

Explain what is meant by this statement.

I. ***(6 marks)***

Discuss the meaning of the following statement: early binding promotes security and efficiency, while late binding promotes flexibility and expressiveness. Do you agree or disagree? Why? Give examples which support your answer. Make sure to consider both facets of the statement in your discussion.

J. ***(6 marks)***

In Gofer, one can give fewer arguments to a function than is expected. I.e. if a function takes 3 arguments, one can supply only one argument, or just two.

Now consider the case of Prolog. One might have a predicate with 3 arguments. Queries involving the predicate can have all three arguments instantiated (i.e. given). However, queries involving the predicate can also have some or all of the arguments left uninstantiated (i.e. as variables).

Therefore, with both Prolog and Gofer, one can construct computations involving "less input than expected". Compare and contrast the two scenarios (in Prolog and Gofer). I.e. how do the two situations or features (of providing "less input than expected") differ conceptually? How are they the same?